Python Assignment Report

Wadkar Srujan Nitin

April 13, 2024

1 Methodology

1.1 Data Preprocessing

The preprocessing steps, EDA done on the data and features extracted are as follows:

- First, an observation was made in the `Constituency` column. `SC` and `ST` categories were present in some rows in parentheses.
- Using this, a new feature was created, namely `Category`, which comprised of `General`, `SC` and `ST` as values. To do this the regex expression was used to get the SC ST values from the given column.
- `Candidate` and `Constituency` columns did not matter as such now. So, they were dropped.
- `Total Assets` and `Liabilities` were converted into numeric values using the **convert_text_to_number** function. In this function, regex expression was used to extract the words such as `Lac`, `Crore`, etc., to get the required values
- For changing the categorical values, two encoding options were available: one hot encoding and label encoding. Both were employed and it was observed that label encoding gave better f1 score than one hot encoding. Hence label encoding was used in the final code for `Category`,`State` and `Party`.
- Scatter plots were plotted for `Total Assets` and `Liabilities`, and outliers were removed by rejecting all the values that were 3 standard deviations away from the mean.
- Techniques such as **SMOTE** were also employed to generate additional data from the given data. But the model performed poorly using the data generated from **SMOTE** as the new data did not accurately corelate with the original data. Hence **SMOTE** was not used in the final code.
- `ID` column was removed from the dataset.
- The pre-processed data was split into train and validation sets in the ratio 4:1, ready to be trained.

2 Experiment Details

The following models were trained on the data:

- Naive Bayes
- Random Forest
- Decision Tree
- Support Vector Machines (Kernel was set to linear)
- Gradient Boosting
- K Nearest Neighbours

All these models were employed after preprocessing the data. The `Naive Bayes` model performed the worst with very low F1 score. Then, `KNN` was used by changing the nearest neighbors' values, but this did not significantly improve the prediction. Changing the models to `Decision Trees` and SVM yielded better results. Finally, the `Random Forest` model was used by setting the `trees` in the forest to 150 and by using `entropy` for information gain in determining the quality of the split. This model performed comparatively better than `SVM` and `Decision Tree` but again, no significant improvement was seen. (0.01)

The libraries used for employing the models and preprocessing were `sklearn`, `scipy`, `numpy`, `pandas`, `matlplotlib`, `seaborn` and others.

2.1 Data Insights

The following insights were obtained from the data:

- Correlation matrix for the given data

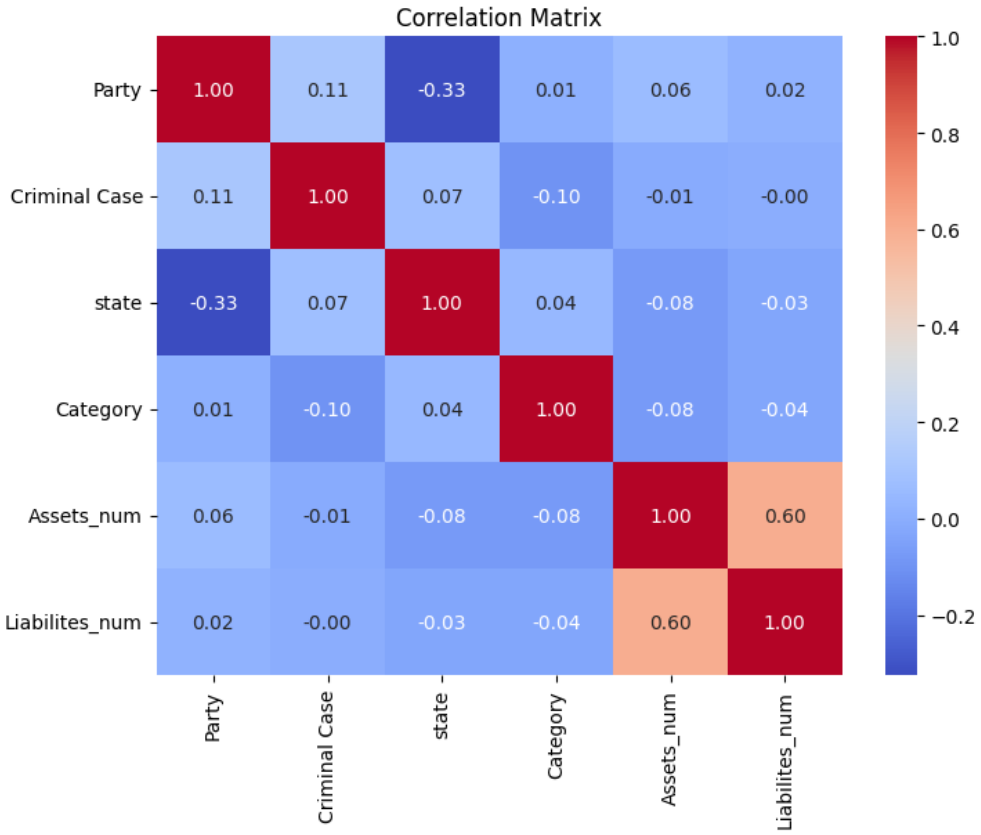

Figure 3: Correlation matrix of the data

We can see here that only Assets and Liabilities have some correlation between them. Rest all columns are pretty much independent of the others and this is the reason why models are not able to perform in a satisfactory manner here

- Scatter plot for Assets and Liabilities before removing outliers

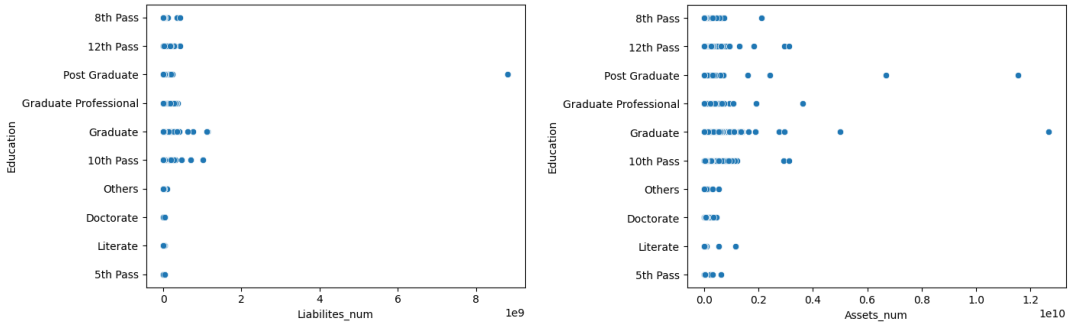

Figure 1: Education vs Assets and Education vs Liabilities before removing outliers

- Scatter plot for Assets and Liabilities after removing outliers

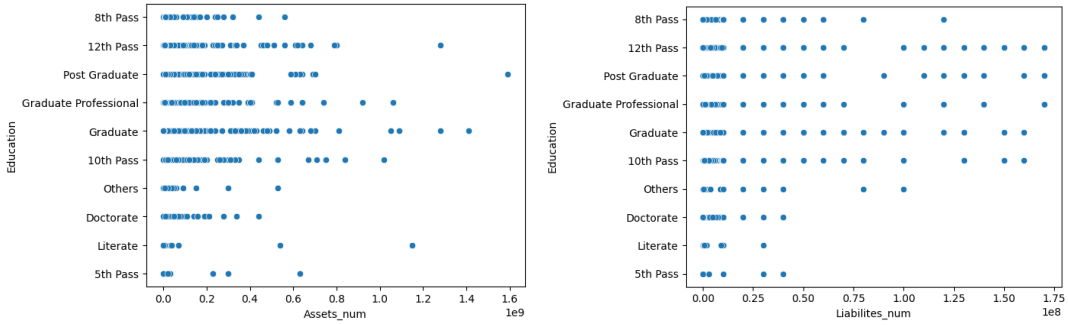

Figure 2: Education vs Assets and Education vs Liabilities after removing outliers

- The percentage distribution of parties with candidates having the most criminal records.

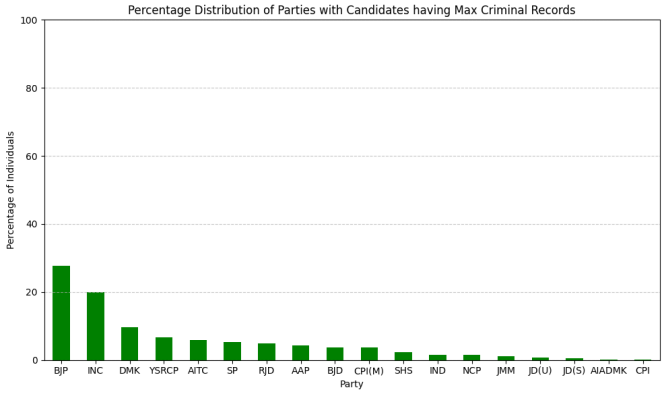

Figure 3: The percentage distribution of parties with candidates having the most criminal records.

- The percentage distribution of parties with the most wealthy candidates.

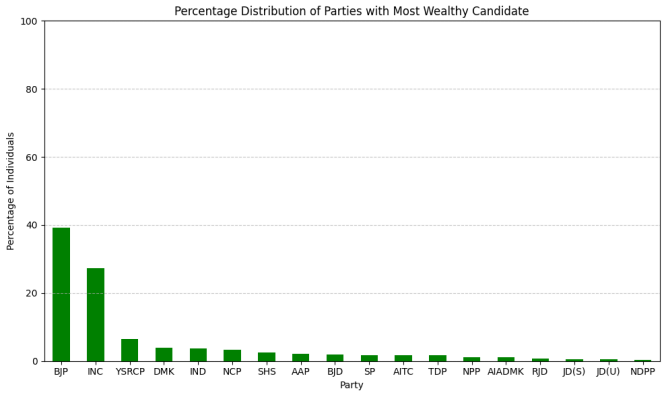

Figure 4: The percentage distribution of parties with the most wealthy candidates.

3 Results

Final F1 score on the test data is as follows:

- Public Score: 0.24238, Rank 91
- Private Score: 0.23473, Rank 101

4 Final Submission

- Link to Github Repo

References

[1] Official documentations of *numpy*, *pandas*, *matplotlib*, *seaborn*, *scikit*, etc.

[2] Medium Blog on Multi Class Classsification